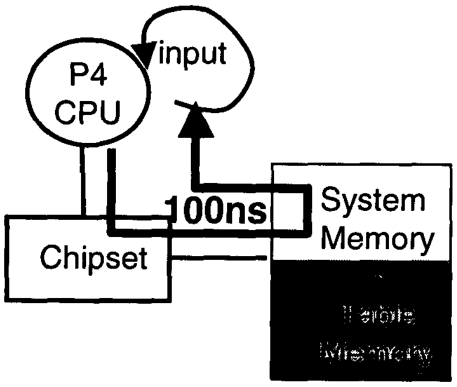
Figure 1(a)

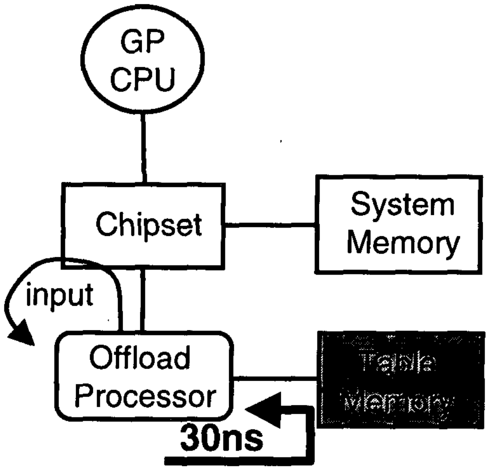
Properties of DFA and NFA techniques used on conventional microprocessors	**Storage: Bound on # of States (for an R character Regular Expression)**	**Evaluation tim (for N bytes of input) [order of]**
Deterministic Finite State Automata or DFA running on a GP CPU	**2^R** (needs very large memory)	**N** memory access cycles
Non-Deterministic Finite State Automata or NFA running on a GP CPU	**R**	**R * N** cpu cache+branch cycles

Figure 1(b)

CPU walking DFA table in DRAM



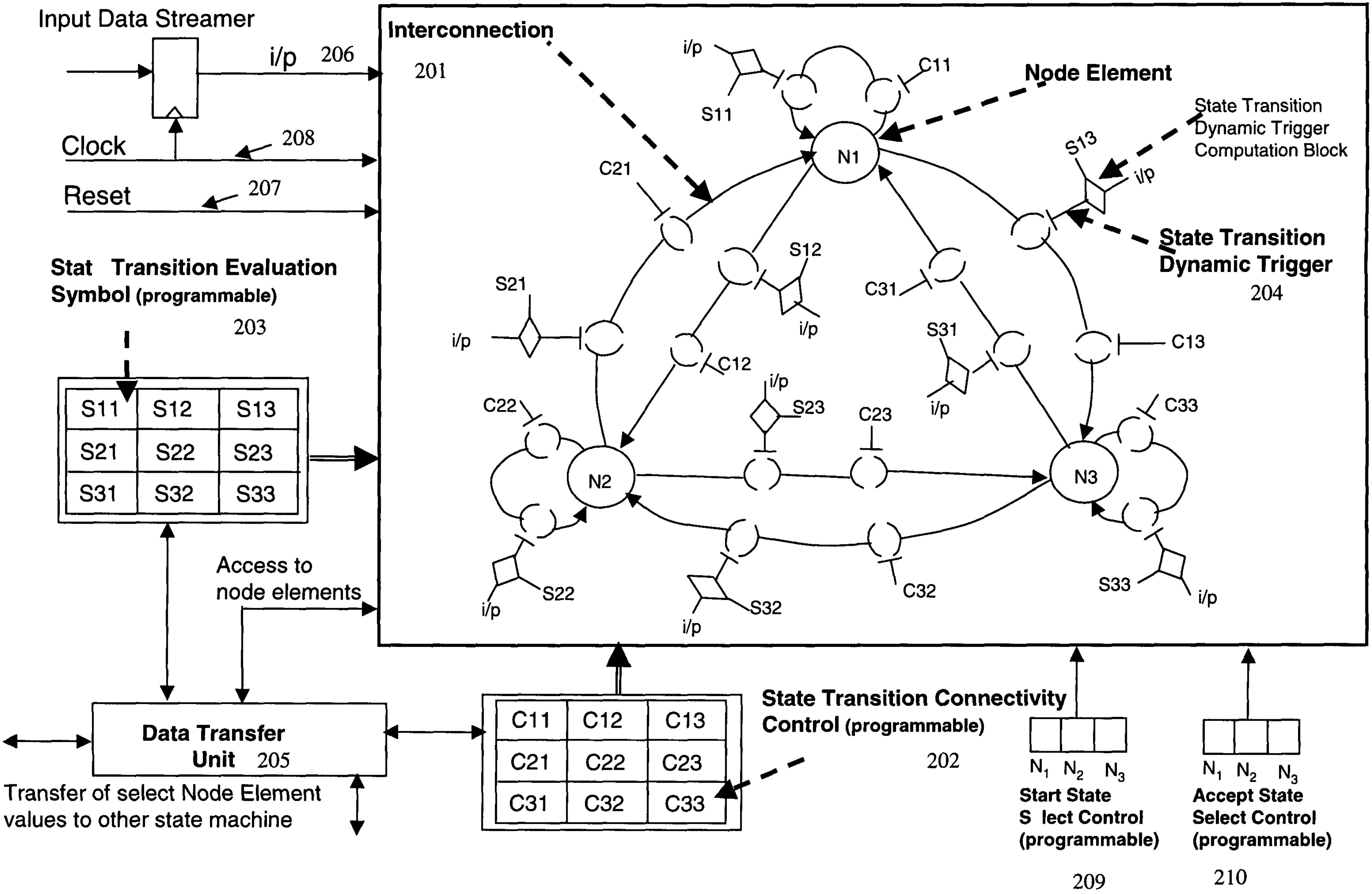
Coprocessor closer to table in SRAM



Performance on evaluating Regular Expressions on every byte of input stream

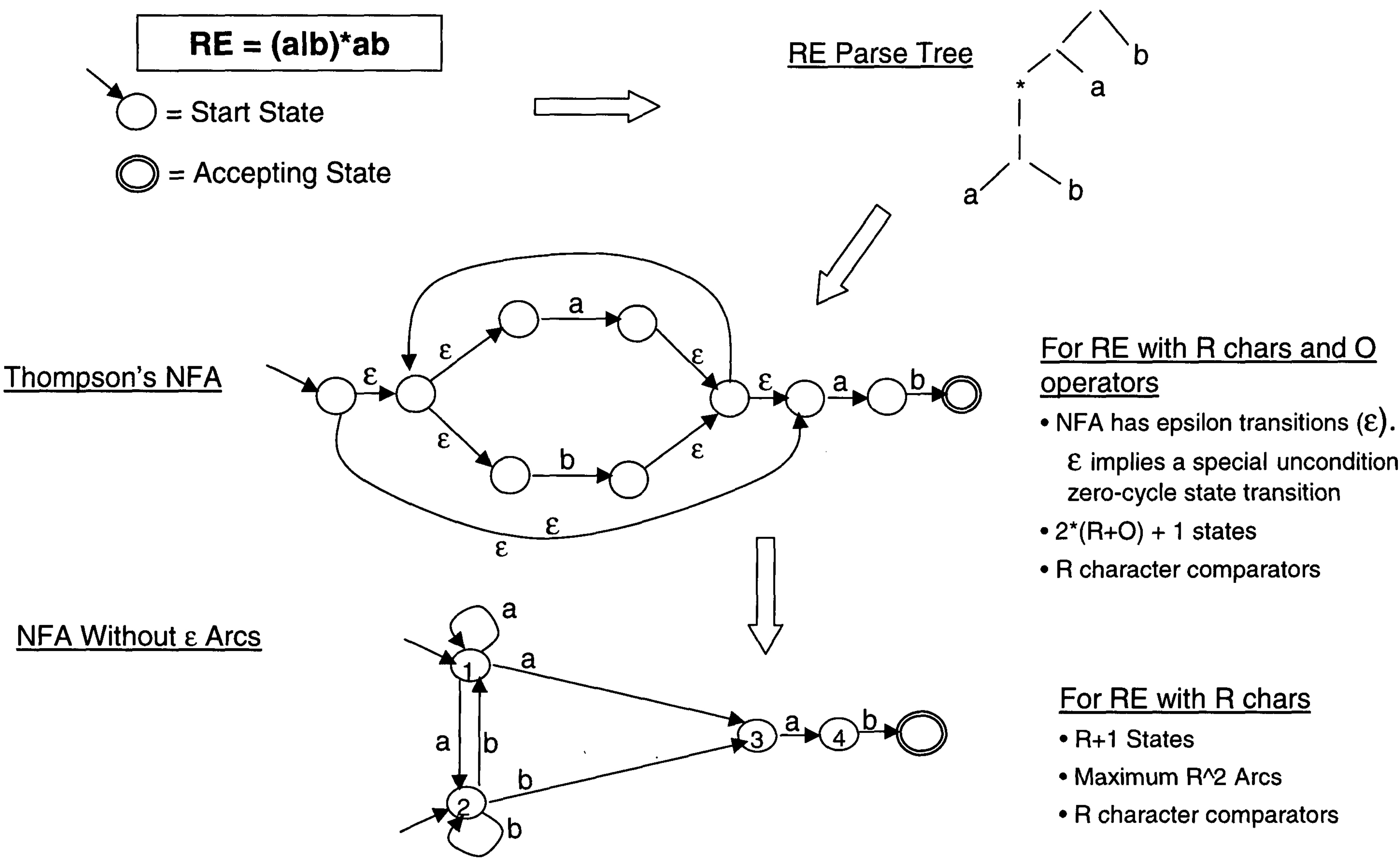
1000s of REs @ 100 Mbps	100s of REs @ 280 Mbps
Gigabytes of Memory	100s of MBs of SRAM

Figure 2



Input Data Streamer
i/p
206
Clock
208
Reset
207
Stat Transition Evaluation
Symbol (programmable)
203
S11
S12
S13
S21
S22
S23
S31
S32
S33
Access to
node elements
Data Transfer
Unit 205
Transfer of select Node Element
values to other state machine
Interconnection
201
Node Element
State Transition
Dynamic Trigger
Computation Block
State Transition
Dynamic Trigger
204
N1
N2
N3
i/p
S11
C11
C21
S12
C31
S13
C13
S21
C12
S31
S23
C23
C22
C33
S22
S32
C32
S33
C11
C12
C13
C21
C22
C23
C31
C32
C33
State Transition Connectivity
Control (programmable)
202
N1 N2 N3
Start State
S lect Control
(programmable)
209
N1 N2 N3
Accept State
Select Control
(programmable)
210

Figure 3(a)



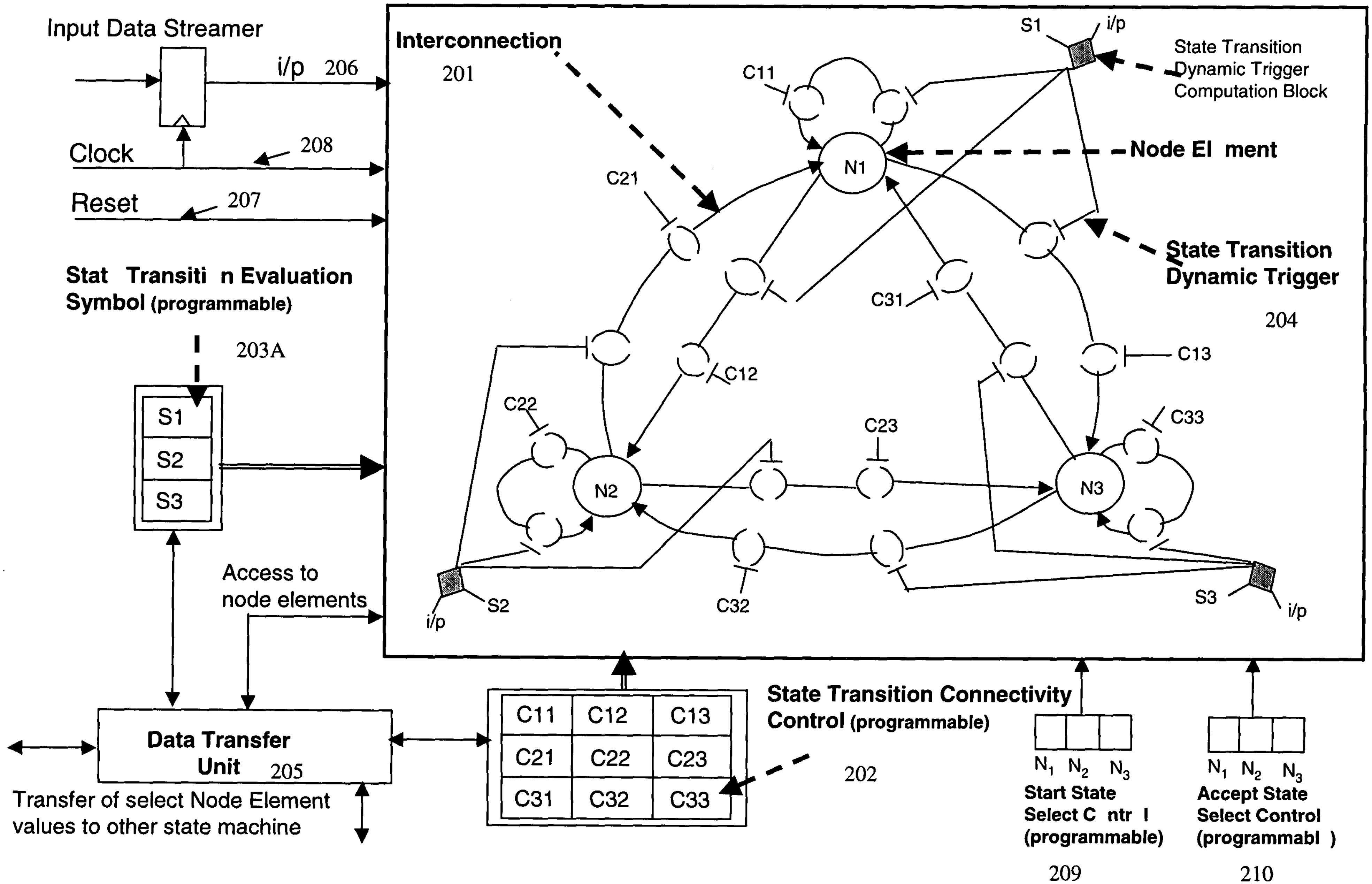
For RE with R chars and O operators

- NFA has epsilon transitions (ε).
  ε implies a special unconditional zero-cycle state transition
- 2*(R+O) + 1 states
- R character comparators

For RE with R chars

- R+1 States
- Maximum R^2 Arcs
- R character comparators

Figure 3(b)



Input Data Streamer
i/p 206
Clock
208
Reset
207
Stat Transiti n Evaluation Symbol (programmable)
203A
S1
S2
S3
Access to node elements
Data Transfer Unit 205
Transfer of select Node Element values to other state machine
Interconnection
201
C11
C21
C12
C22
C23
C31
C13
C33
C32
N1
N2
N3
S1
i/p
S2
i/p
S3
i/p
State Transition Dynamic Trigger Computation Block
Node El ment
State Transition Dynamic Trigger
204
C11
C12
C13
C21
C22
C23
C31
C32
C33
State Transition Connectivity Control (programmable)
202
N1 N2 N3
Start State Select C ntr l (programmable)
209
N1 N2 N3
Accept State Select Control (programmabl)
210

Figure 4

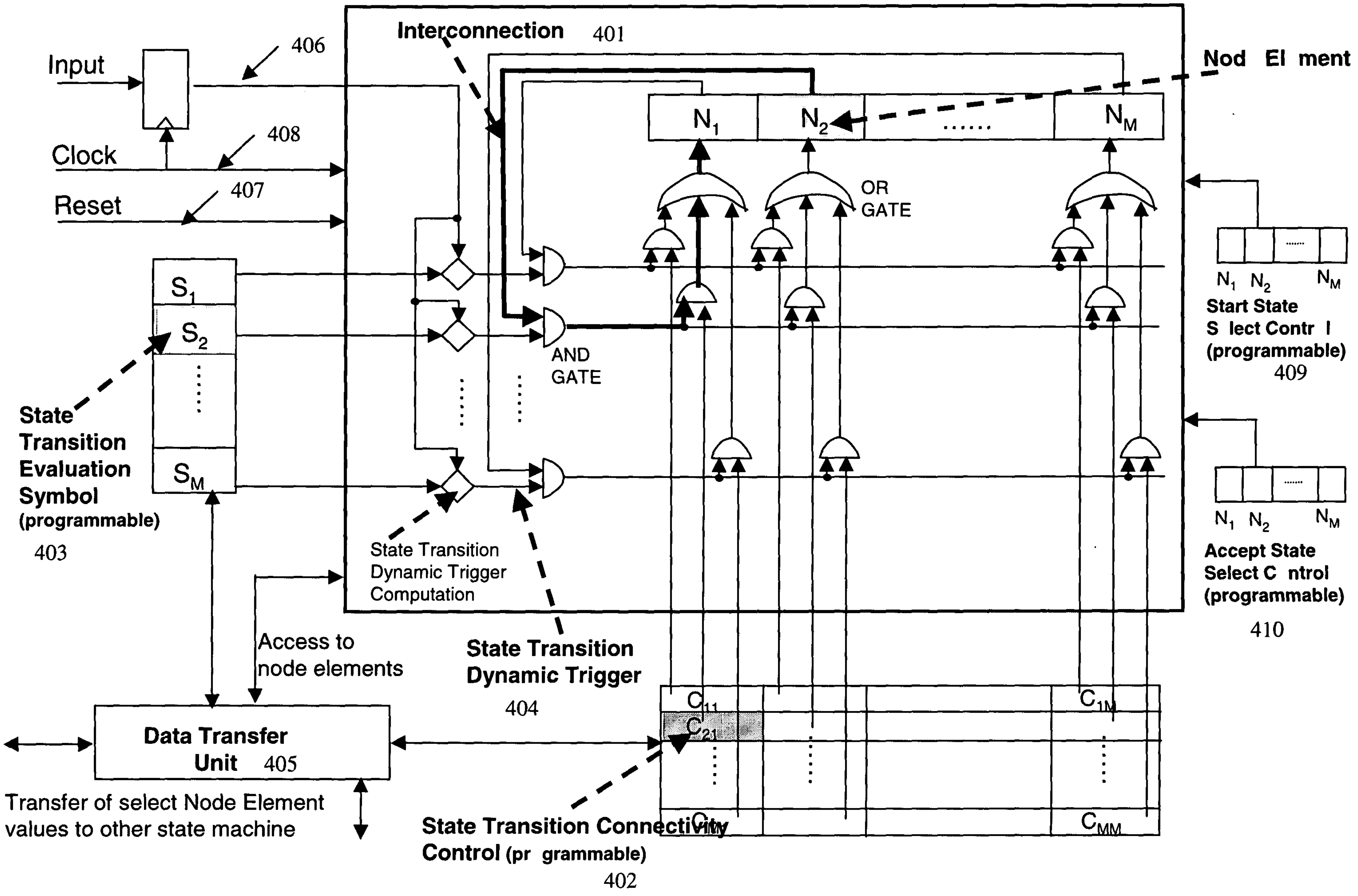

Interconnection 401
Nod El ment
Input
406
Clock
408
Reset
407
S1
S2
SM
N1
N2
NM
OR GATE
AND GATE
State Transition Evaluation Symbol (programmable)
403
State Transition Dynamic Trigger Computation
State Transition Dynamic Trigger
404
Access to node elements
Data Transfer Unit 405
Transfer of select Node Element values to other state machine
C11
C21
C1M
CMM
State Transition Connectivity Control (pr grammable)
402
N1 N2 NM
Start State S lect Contr l (programmable)
409
N1 N2 NM
Accept State Select C ntrol (programmable)
410

Figure 5

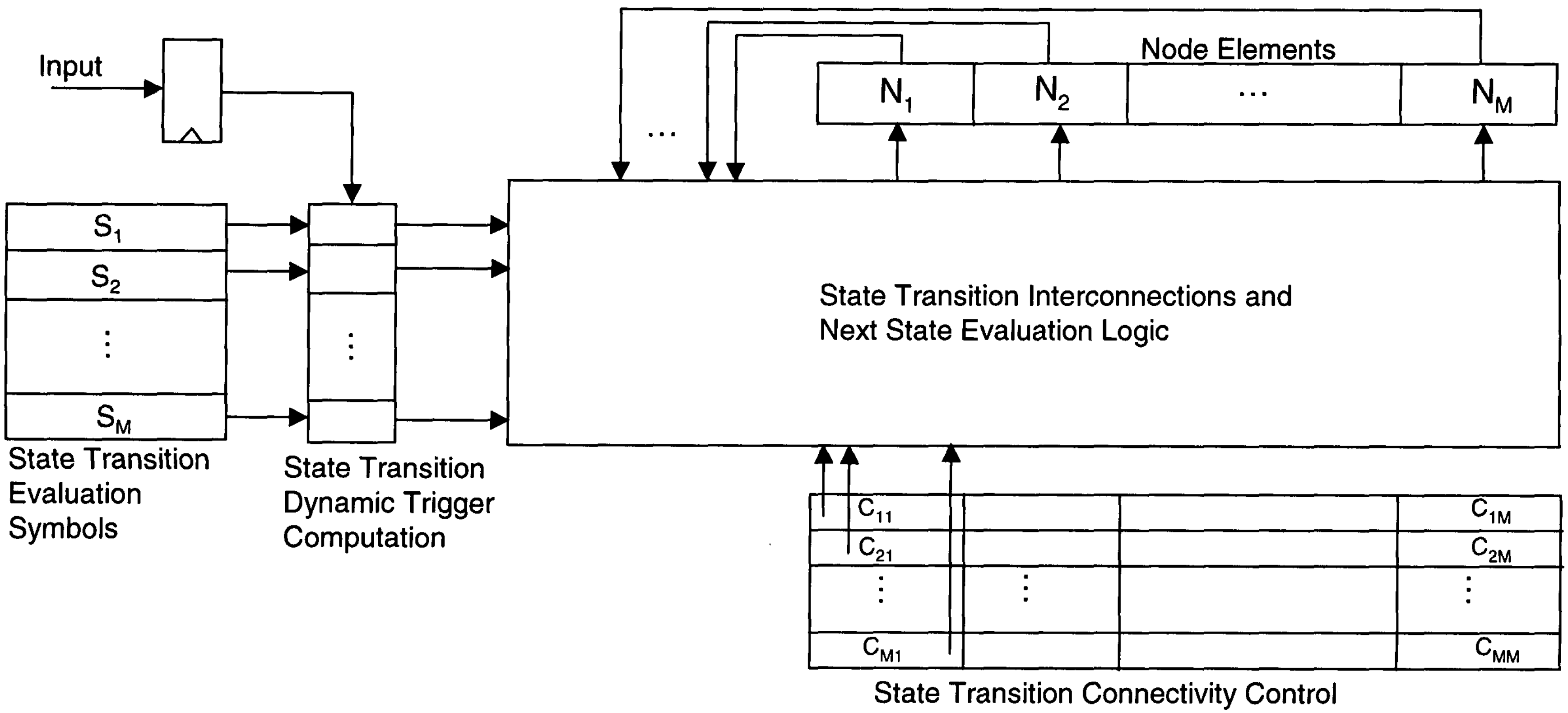
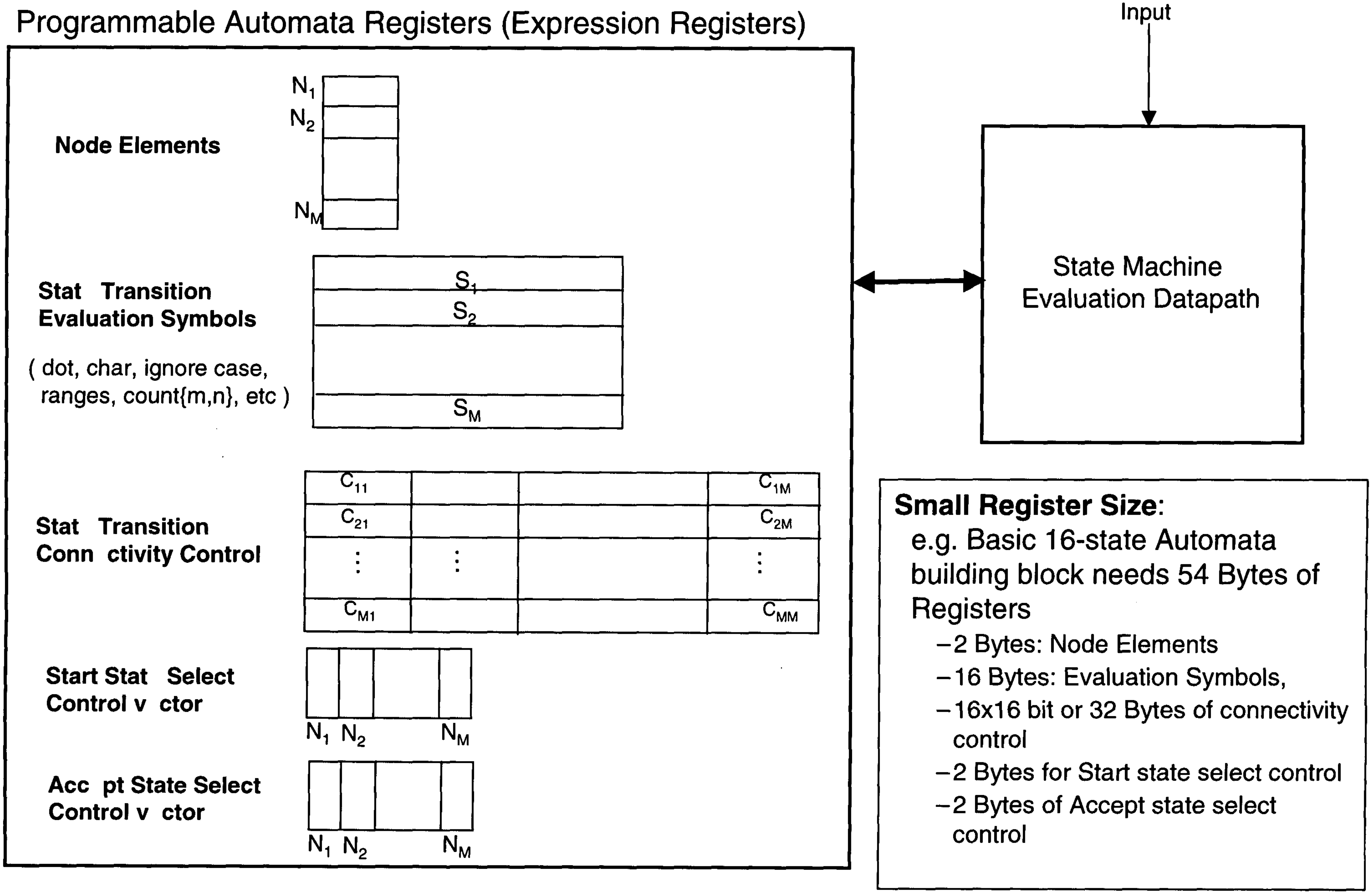

Input
Node Elements
N_1
N_2
...
N_M
...
S_1
S_2
⋮
S_M
State Transition Evaluation Symbols
⋮
State Transition Dynamic Trigger Computation
State Transition Interconnections and Next State Evaluation Logic
C_{11}
C_{21}
⋮
C_{M1}
⋮
C_{1M}
C_{2M}
⋮
C_{MM}
State Transition Connectivity Control

Figure 6


Programmable Automata Registers (Expression Registers)
Input
Node Elements
N_1
N_2
N_M
Stat Transition Evaluation Symbols
(dot, char, ignore case, ranges, count{m,n}, etc)
S_1
S_2
S_M
Stat Transition Conn ctivity Control
C_{11}
C_{1M}
C_{21}
C_{2M}
C_{M1}
C_{MM}
Start Stat Select Control v ctor
N_1 N_2 N_M
Acc pt State Select Control v ctor
N_1 N_2 N_M
State Machine Evaluation Datapath

Small Register Size:

e.g. Basic 16-state Automata building block needs 54 Bytes of Registers

- –2 Bytes: Node Elements
- –16 Bytes: Evaluation Symbols,
- –16x16 bit or 32 Bytes of connectivity control
- –2 Bytes for Start state select control
- –2 Bytes of Accept state select control

Fig 7

- Simple regular structure enables a high density → dense array of multiple tiles

  Several thousands of automata (organized as multiple rows of tiles) can fit on a singe die on 0.13u technology

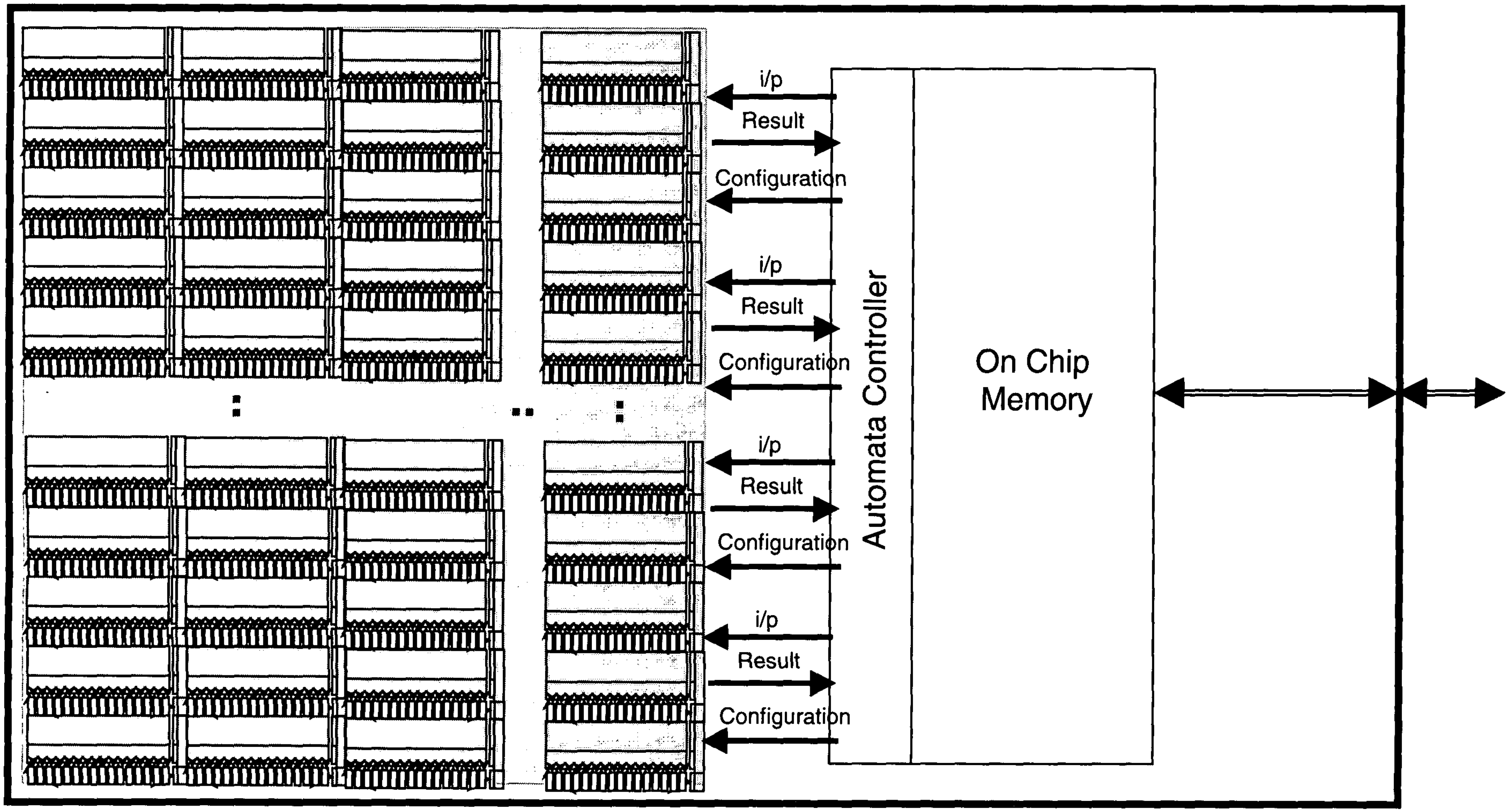


Figure 8(a)

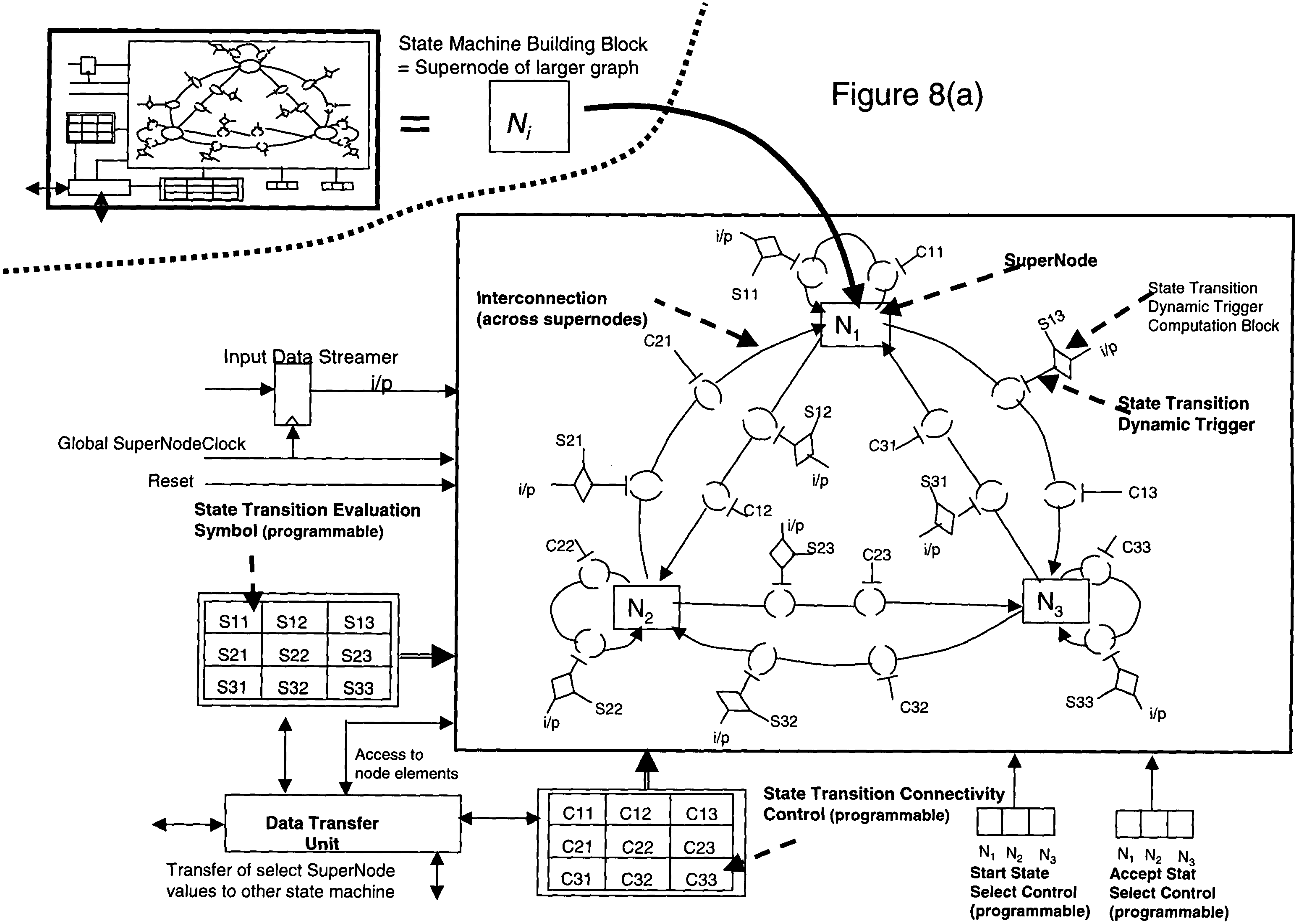
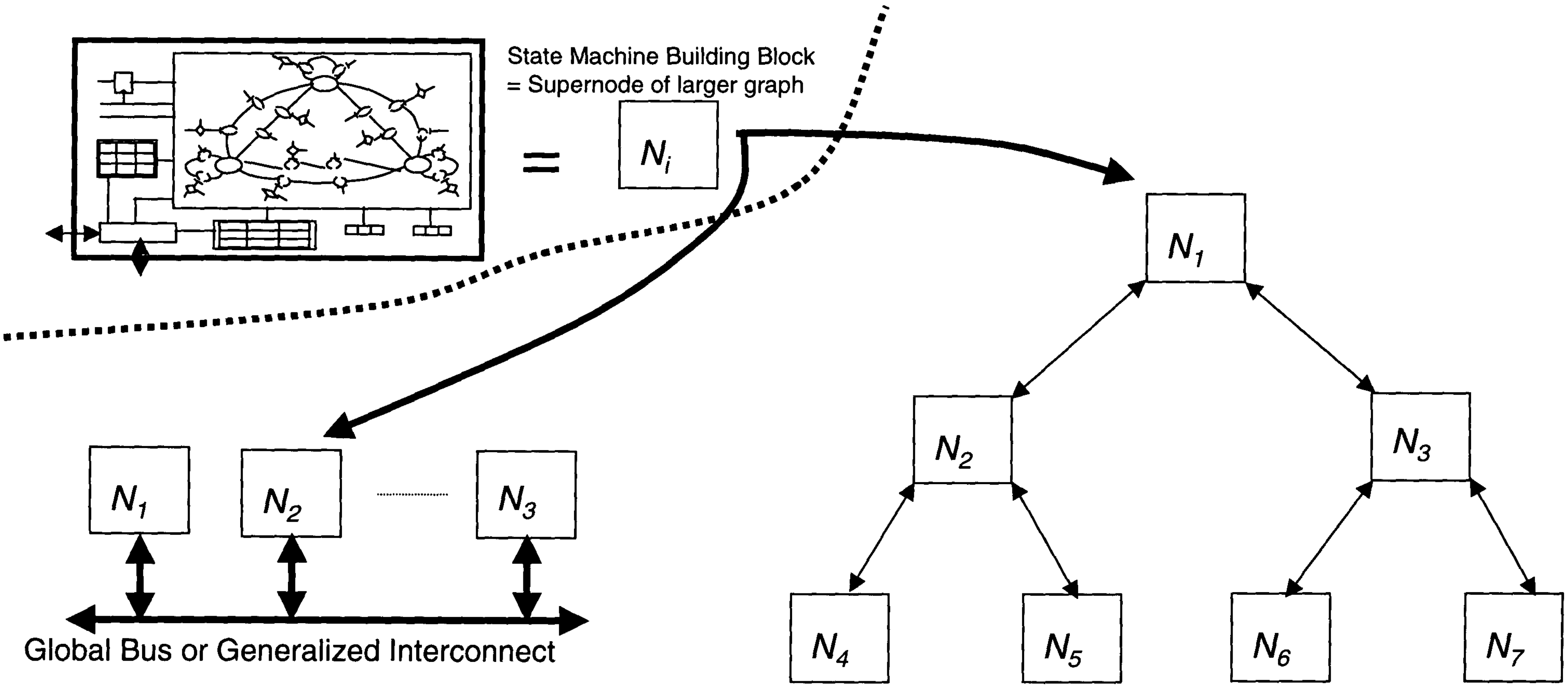

State Machine Building Block
= Supernode of larger graph
=
N_i
SuperNode
Interconnection
(across supernodes)
State Transition
Dynamic Trigger
Computation Block
State Transition
Dynamic Trigger
N_1
N_2
N_3
i/p
S11
C11
C21
S12
C31
S13
S21
C12
S31
C13
C22
S23
C23
C33
S22
S32
C32
S33
Input Data Streamer
i/p
Global SuperNodeClock
Reset
State Transition Evaluation
Symbol (programmable)
S11
S12
S13
S21
S22
S23
S31
S32
S33
Access to
node elements
Data Transfer
Unit
Transfer of select SuperNode
values to other state machine
C11
C12
C13
C21
C22
C23
C31
C32
C33
State Transition Connectivity
Control (programmable)
N_1 N_2 N_3
Start State
Select Control
(programmable)
N_1 N_2 N_3
Accept Stat
Select Control
(programmable)

Figure 8(b)


State Machine Building Block
= Supernode of larger graph
=
N_i
N_1
N_2
N_3
Global Bus or Generalized Interconnect
N_1
N_2
N_3
N_4
N_5
N_6
N_7

Larger Stat Machine Realized via Multiple State Machine Building Blocks on a Gen ral Interconnect

Larger State Machine Realized via Multiple State Machine Building Blocks Organized as a Tree

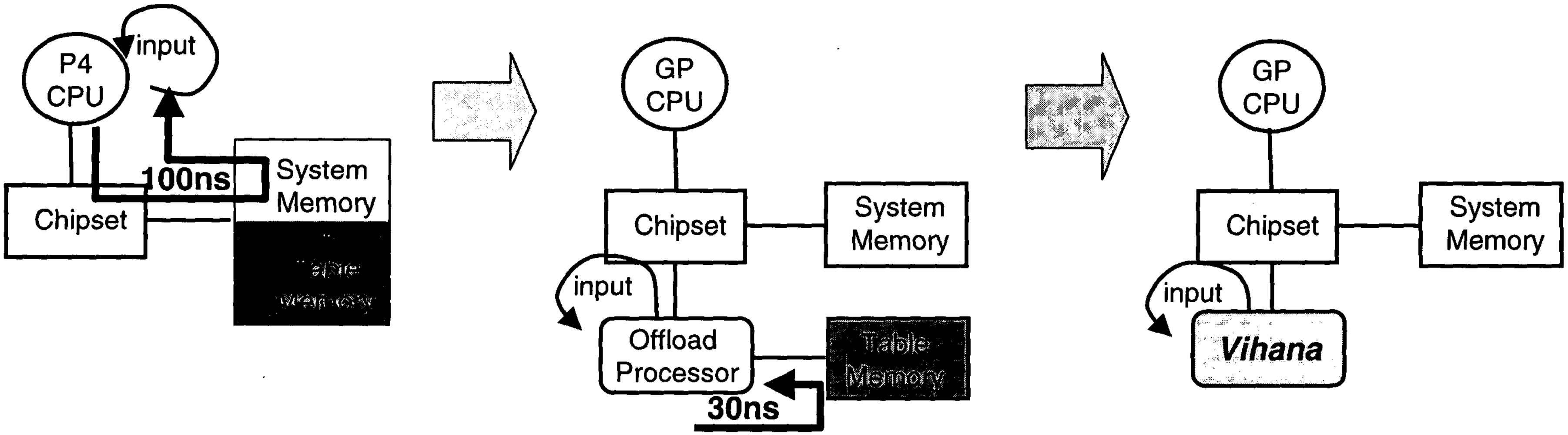
Figure 9(a)

Prior Art

Properties of DFA and NFA techniques used on conventional microprocessors	**Storage: Bound on # of States (for R characters)**	**Evaluation time (for N bytes) [order of]**
D terministic Finite State Automata or DFA running on a GP CPU	**2^R** (needs very large memory)	**N** memory access cycles (~100ns)
Non-Deterministic Finite State Automata or NFA running on a GP CPU	**R**	**R * N** cpu cache+branch cycles (~4ns)
Non-Deterministic Finite State Automata r NFA running on the Apparatus	**R**	**N** Tight on chip state transition cycle (~1 ns)

Figure 9(b)

CPU walking DFA table in DRAM	**Coprocessor closer to table in SRAM**	**Regular Expression Co-processor using Exemplary State machine Architecture**



Perf on REs on every byte

1000s of REs @ 100Mbps	100s of REs @ 280Mbps	**1000s of REs @ > 10Gbps**
Gigabytes of Memory	100s of MBs of SRAM	**No table memory needed**

Two orders of magnitude speedup without need for table memory